LA-UR-15-27748

Approved for public release; distribution is unlimited.

Title: Generating Cinema Databases for In Situ Visualization of Ocean Modeling Simulations

Author(s): Eatmon, Arnold
Canada, Curtis Vincent
Patchett, John M.

Intended for: Web

Issued: 2015-10-05

Generating Cinema Databases for In Situ Visualization of Ocean Modeling Simulations

Arnold Eatmon
Fort Valley State University

Curt Canada
Los Alamos National Laboratory

John Patchett
Los Alamos National Laboratory

ABSTRACT

Science is changing. In the last few years science has seen a dramatic shift towards data intensive discovery, a combination of past paradigms of discovery integrated with computational power and an ample supply of data from which we can derive information.

One notable problem with this is that while data and computational power are increasing, storage is decreasing. Storage in this day and age is a resource, and resources are inherently limited. Due to being a resource decisions must be made on how to wisely utilize storage to tackle scientific challenges.

Cinema databases allow for in situ processing and visualization, eliminating the need to write large amounts of data to disk. Cinema databases allow for scientists to not only view the data but also to interact with the data in meaningful ways. Simultaneously, cinema databases drastically reduce the amount of storage utilized in simulation.

In this project, I applied cinema database technology to a climate simulation model, MPAS-Ocean.

Author Keywords

Computer Science; Earth Science; Climate Change; Climate Modeling; Simulation; In Situ Visualization; MPAS-O; Cinema

I. BACKGROUND

MPAS-Ocean

MPAS is a climate modeling simulation developed at Los Alamos National Laboratory. The name is an abbreviation for Model for Prediction Across Scales. The purpose of MPAS is to aid scientists and decision makers by utilizing simulation to inform on climate change over time.

MPAS is designed and managed by the LANL Climate Ocean and Sea Ice Modeling (COSIM) Team. The currently released components of MPAS are the Land Ice, Atmosphere, and Ocean components.

Cinema

Cinema is an In Situ visualization and analysis tool; it allows for visualization during the simulation and decreases the storage space needed to utilize simulation data.

Unless scientists know what they want in advance, extreme scale science requires writing large amounts of data to disk for analysis. Cinema allows scientists to interact with data in a compact way, drastically decreasing disk space used in the input/output process.

Cinema works by capturing images as the simulation runs in place at different angles and time steps then creating an interactive visualization that can be used to make decisions and identify regions of interest.

II. RESEARCH METHODOLOGY

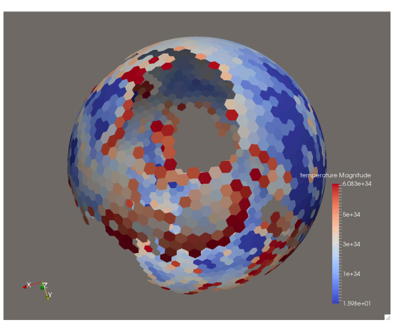
Software Building

The initial build process involved getting the prerequisites for MPAS-O up and running on the HPC. We initially used Darwin here at LANL to build software. However, we encountered a security issue that did not allow code to send and receive information out to the internet, processes like cloning git repositories did not work as a result. To remedy this issue, we made a switch to a LANL HPC on the Turquoise network called Hobo.

Hobo was able to receive an environment set up for MPAS-Ocean by LANL scientist David Rogers on LANL's mustang HPC. We utilized this build environment, copying the directory structure and creating an alias called "cwork" to access the environment, to proceed with our work in MPAS-O.

Initial Cinema Databases

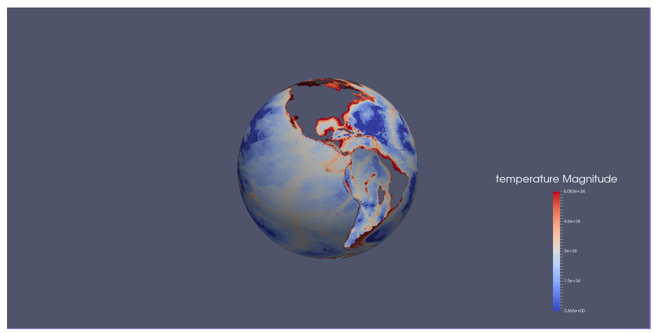
The initial cinema databases were generated at 240 km resolution using a file generated via Coprocessing called sphere_temp_20x20.py, however this resolution is a grid used for test not for visualization as it does not show discernible detail.



The same python file worked without issue in the directory structure set up for 60 km resolution and output a cinema database. However, several issues persisted with these databases even at a discernible resoltution.

Despite being at a higher, 60 km resolution these were still relatively low resolution as COSIM scientists were currently doing full visualizations in 15 km resolution and working towards detail in 10 km resolution during the course of this research project.

Beyond the low resolution of the databases, there was not a known way to alter the colormap of the cinema databases. As a result, while domain scientist were in practice using visualizations that had colormaps designed by LANL visual artists Francesca Samsel, intended to intensify the visual perception of detail in MPAS-O simulations, the cinema databases were in the default Moreland Cool-Warm color map.



The default simulation time was set for a year at an interval of once per week data retrieval, when in practice climate change scientist are interested in simulations decades into the future.

Cinema databases initially could only be generated for the variable temperature. However, scientists are interested in visualizations of many variables. These variables included Salinity, Relative Vorticity, Kinetic Energy, and Okubo Weiss.

The generation of cinema databases was slower than necessary; this was due to being set to run on a single node with 16 processors per node. However, the simulations could be run with various amounts of processors in practice, dictated by grid files that differ in each resolution.

Finally, the initial cinema databases did not contain useful, insightful information. For the purpose of illustration, the upper limit of the legend is a value of 6.083e34 degrees Celcius. The values did not reflect actual ocean temperatures. The issue was one not only residing in the data range chosen for the legend but also one where the simulation utilized the entire vertical coordinate as a vector and computing the temperature as the magnitude of that whole vector rather than for a single layer.

Generation at Higher Resolutions

The first challenge was to improve the resolution of the cinema databases to current standards of 15 km. In order to do this, a directory structure was created by copying the current directory structure for lower resolutions and removing the graphs directory, restarts directory, restart timestamp file, namelist.ocean_forward file, and streams.ocean_forward file. These were replaced with those on the mpas development website for 15 km resolution. This resulted in several errors that were difficult to pin point the solution to as they were specific to MPAS-O functionality.

MPI errors of various specifics were eventually resolved by identifying that certain components of the MPAS-O simulation were not defined in input files for higher resolutions. This issue was resolved by adding the following lines to the below specified files. To streams.ocean_forward, the following lines needed to be added in the 15 km resolution file:

```
<stream name="okuboWeissOutput"
        type="output"

filename_template="analysis_members/okuboWei
ss.$Y-$M-$D_$h.$m.$s.nc"
        filename_interval="01-00-
00_00:00:00"
        packages="amOkuboWeiss"
        clobber_mode="truncate"
        output_interval="0001_00:00:00">

    <stream name="mesh"/>
    <var name="xtime"/>
    <var name="okuboWeiss"/>
    <var name="eddyID"/>

</stream>

<stream name="paraview_catalystOutput"
        type="output"

filename_template="analysis_members/paraview
_catalyst.$Y-$M-$D.nc"
        filename_interval="01-00-
00_00:00:00"
        packages="paraview_catalyst"
        clobber_mode="truncate"
        output_interval="00-00-07_00:00:00">
    <var name="xtime"/>
```

To the file namelist.ocean_forward in the 15 km and like resolutions, the following lines needed to be added to the end of the file:

```
&okubo_weiss
    config_use_okubo_weiss = .true.
    config_okubo_weiss_compute_startup =
.true.
    config_okubo_weiss_directory =
'analysis_members'
    config_okubo_weiss_threshold_value = -
0.2
    config_okubo_weiss_normalization = 1e-10
    config_okubo_weiss_lambda2_normalization
= 1e-10
    config_okubo_weiss_use_lat_lon_coords =
.true.
    config_okubo_weiss_compute_eddy_census =
.true.
    config_okubo_weiss_eddy_min_cells = 100
/

&paraview_catalyst
    config_use_paraview_catalyst = .true.

config_paraview_catalyst_compute_interval =
'000-00-07_00:00:00'
    config_paraview_catalyst_compute_startup
= .true.
    config_longitude_periodic_split = 180.
```

In order to generate these higher resolution databases, the number of processors used needed to be increased. It’s necessary in MPAS-O functionality for the number of processors used to match the number of tasks in a grid file, grids changing with resolution. While the 60 km resolution contained files that could use processors of the amounts 16, 64, 128, 256, and 512, the 15 km resolution used processors in the amounts 128, 256, 512, 1024, 2048, and 4096. As more processors are used, the simulation runs more quickly. As a result, 1024 processors on the Hobo HPC allowed for a better run speed.

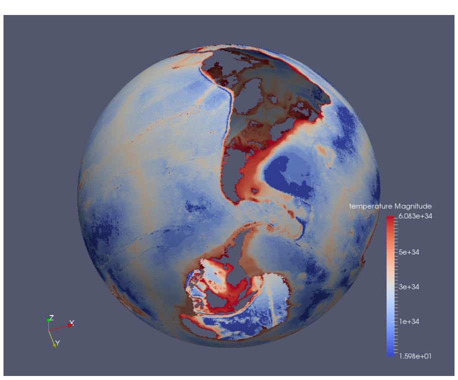
An error arose as the amount of processors was increased. The error reads “a process failed to create a queue pair” and list possible reasons being insufficient memory or that no more memory can be registered with the device. This error is resolved by utilizing the bind to core option for runs at a higher resolution using more processors. In the mpirun command, the line is altered to the following:

```
mpirun --bind-to-core –np $numproc ocean_forward_model
```

Also, set the environment variable I_MPI_DAPL_UD to enable by adding the following line to the run file or by declaring it in the command line prior to the mpirun command if running interactively:

```
setenv I_MPI_DAPL_UD enable
```

These changes allowed for the generation of cinema databases at a resolution of 15 km.



MPAS-O Simulation Time Controls

MPAS controls are defined in great detail in the user guide. However, some functions are worth noting here for the convenience of use and brevity.

Time steps are units of time in a simulation and in MPAS-O are roughly equal to the total simulated time divided by the interval at which information is collected in the simulation.

For illustration, a simulation with a length of one day that collects information and an interval of one day will have a single time step. Because the amount of time steps is controlled by multiple factors in the MPAS-O code, this is not always a direct ratio especially as the difference in simulated time to interval of collection increases. However it is still a good approximation.

The length of simulated time and the interval at which information is collected are controlled in the input file namelist.ocean_forward of any given run directory. They can be altered by opening the file in a text editor and altering the config_run_duration variable and the config_stats_interval variable located on the 6th and 11th lines of the file respectively.

The format of the config_stats_duration file is Years-Months-Days_Hours:Min:Sec . The format of the config_stats_interval format is Days_Hours:Min:Sec. The settings for the prior one time step example look like:

```
config_run_duration='0000-00-01_00:00:00'

config_stats_interval = '0001_00:00:00'
```

Initial Methods to Import Color Maps

The initial method used to enter color maps was to enter in the RGB values manually into the python script. This is done by identifying the RGB values for the color control points of a particular color map and entering them in an RGB triplet format. An RGB triplet contains values in the format Scalar, R, G, B, Scalar, and so on. While the Moreland cool-warm color map contains three color control points, the resident visual artists' color maps had over thirty making manual entry, while doable, quite cumbersome.

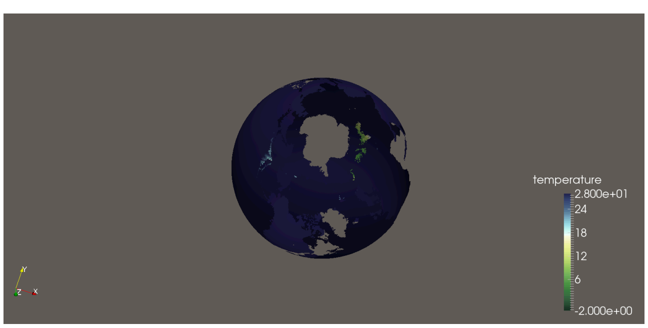
This problem was later remedied by using the trace function in ParaView to attain the color map formatted to an RGB triplet in python code. However this still required interpolation from Lab space, where the color maps were designed, to RGB space which would occasionally result in an out of place color in the legend.

Initial Adjustment of Data Ranges

Similar to the entry of color maps, changing the data range first had to be done manually. This was done by taking the range of interest in the data and subtracting the minimum value from the maximum value, then dividing by the number of color control points in a particular color map. Then, to find evenly spaced scalars, the result of this equation is added recursively starting from the minimum value until the maximum value is reached.

A major disadvantage to this approach is that each change in color maps requires the data range to be redone. In the same vein, errors in rounding can result in uneven spacing in the color map.

Altering the data range and color map, while a step forward, did not resolve issues stemming from the simulation using the vertical vector's magnitude for temperature.

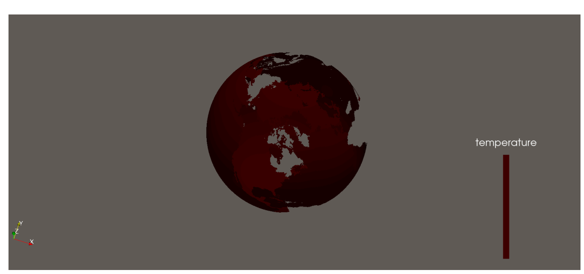


Issues in the Generation of New Scripts

The generation of new scripts led to a couple of new errors in the process of improving the workflow of the generation of cinema databases. Two notable issues were the lack of a recognized variable for input due to mixed versions of ParaView and undeclared visualization properties.

In ParaView the window for variable selection is Cell/Point Array Status. In the code, this is reflected as a call for either CELLS or POINTS. Incorrectly calling one when the other is required as well as the absence of a ColorBy function declared following the statement resulted in output cinema databases that lacked any coloring. Using different versions of ParaView for exporting a script versus writing out cinema databases further contributes to this issue due to differences in plugins, patches, and such that aren't recognized.

The absence of declarations of some settings in ParaView can also cause issue. By default some values of most settings are 0, however the interaction of some functions changes these to 1. The following cinema output resulted from making alterations in the python code to the ColorBy function without explicitly declaring Use log scale and Discretize, which should be set to 0 and 1 respectively.



III. FIANLIZED WORKFLOW

Building MPAS-Ocean

MPAS-O source code is publicly available at the URL http://mpas-dev.github.io/. Prior to compilation, other components that must be downloaded and compiled are NetCDF (serial), Parallel NetCDF, and Parallel Input/Output (PIO). Build instructions are outlined in the Prerequisites section of the MPAS-O User Manual which can be found at this hyperlinked LANL website.

Detailed build instructions for the LANL turquoise network HPC Hobo can be found as outlined by LANL Scientists David Rogers at the following URL

https://darwin.lanl.gov/projects/vizproject/wiki/Cinema-studio

For users with access to LANL's Hobo HPC, there are instructions to build an environment for MPAS-O with Cinema functionality on the above website as well.

Attaining ParaView Coupled with Cinema

Using a version of ParaView that is coupled with cinema and has an enabled Coprocessing Plugin drastically simplifies the process and is highly recommended as it is the workflow cinema is designed to run from.

The version of ParaView used in this paper is 4.3.1-928-gfc4ca77 64-bit. For those compiling ParaView without access to the version in LANL HPC's that is coupled with MPAS in the build from LANL Scientist David Rogers, the following is additional information regarding the configurations:

Version	4.3.1-928-gfc4ca77 64-bit
Qt Version	4.8.6
Architecture	64
Embedded Python	On
Python Library Path	/System/Library/Frameworks/Python.framework/…
Python Library Version	2.7.6 (default, Sep 9 2014, 15:04:36) [GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-6…
Python Numpy Support	On
Python Numpy Path	/System/Library/Frameworks/Python.framework/…
Python Numpy Version	1.8.0rc1
Python Matplotlib Support	On
Python Matplotlib Path	/System/Library/Frameworks/Python.framework/…
Python Matplotlib Version	1.3.1
Python Testing	Off
MPI Enabled	Off
Disable Registry	Off
Test Directory	
Data Directory	
OpenGL Vendor	ATI Technologies Inc.
OpenGL Version	2.1 ATI-1.32.24
OpenGL Renderer	AMD Radeon R9 M290X OpenGL Engine

Attaining Adaptor Files

In order to get a sort of template from which you can use in Paraview for visualization, you need output that matches the input found in the adaptor of MPAS-Ocean. There are ten different kinds of adaptor input files, which are:

```
X_Y_NLAYER-primal
X_Y_NLAYER-dual
X_Y_Z_1LAYER-primal
X_Y_Z_1LAYER-dual
X_Y_Z_NLAYER-primal
X_Y_Z_NLAYER-dual
LON_LAT_1LAYER-primal
LON_LAT_1LAYER-dual
LON_LAT_NLAYER-primal
LON_LAT_NLAYER-dual
```

A script can be written or generated to produce all ten types of output, labeled accordingly with VTK unstructured extensions following. If you know you will be using one specific type of adaptor file, you are able to utilize one of the existing python files in the MPAS-O directory to generate that specific type of adaptor file.

Included in my archive is a file generated by Kitware labeled mpasgridwriter.py which will generate all ten types of adaptor files. Utilize it by soft linking to mpas.py, then adjusting the time in the namelist.ocean_forward file to one day at an interval of one day.

The resolution that these adaptor files are generated in does not need to match the resolution of the cinema database they will be used to generate. If you want your end result to be a cinema database in 15 km resolution, it is perfectly fine to generate adaptor output in 60 km resolution.

However, it is not recommended to use the 240 km resolution to produce adaptor files. The 240 km resolution is not used for visualization, rather it is simply a test. Utilizing the 240 km will make visualization in Paraview quite difficult as the resolution makes it hard to identify features that may be obscured by choices in data range or color map; higher resolutions more accurately represent your final result.

The advantage to utilizing a lower resolution than you need to generate the adaptor files is that the generation of these files can be done quicker while using fewer processors. For example, the generation of all ten types of adaptor output for a resolution of 15 km can take over 15 minutes using 128 nodes with 16 ppn. The same amount of adaptor output can be attained in the 60 km resolution with 32 nodes and 16 ppn in just under 2 minutes.

Using the mpasgridwriter.py should go along the lines of the following:

```
$ cd mpas_060km/
$ ln -s -f mpasgridwriter.py mpas.py
$ cd ../
$ msub msub_060km
```

These commands do not include changing the simulated time, which is addressed elsewhere in this paper, nor do they include the option to change the amount of processors used, which is also addressed in another section of this paper.

If you know you are interested in a single type of adaptor file from the get go, another option is to utilize the mpas_vtkwriter.py file that is included in the MPAS-O build directory. To utilize this file, open it in the text editor of your choice and the scroll to a section of the code that looks like the following:

```
datasets = {
 'MPAS_OUTPUT': {
 # the grid to output
 #'grid': 'X_Y_NLAYER-primal',
 #'grid': 'X_Y_NLAYER-dual',
 #'grid': 'X_Y_Z_1LAYER-primal',
 #'grid': 'X_Y_Z_1LAYER-dual',
 #'grid': 'X_Y_Z_NLAYER-primal',
 #'grid': 'X_Y_Z_NLAYER-dual',
 #'grid': 'LON_LAT_1LAYER-primal',
 #'grid': 'LON_LAT_1LAYER-dual',
 #'grid': 'LON_LAT_NLAYER-primal',
 #'grid': 'LON_LAT_NLAYER-dual',
```

Simply navigate to the option for the adaptor format you wish to use and remove the # symbol in front of that line. If you desire more than one type of output, remove the # symbol from both types of input. In the cinema databases I generated, I chose to utilize X_Y_Z1LAYER-primal.

Another option available while using the mpas_vtkwriter.py file is to select multiple variables for output in the adaptor files. This is done by altering the following line of code:

```
# fields to output
fields':['okuboWeiss','temperature','kin
eticEnergyCell','relativeVorticityCell’]
```

Substituting or adding another variable is possible in this line, provided that variable is defined in code. For example, if you wanted output that included salinity but not Okubo Weiss, the field would show as follows:

```
# fields to output
fields':['salinity','temperature','kinet
icEnergyCell','relativeVorticityCell’]
```

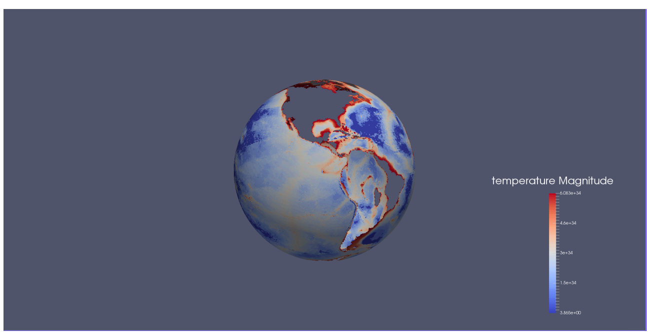
This file is made use of in using the same code lines that would be used for mpasgridwriter.py, simply swapping the name to mpas_vtkwriter.py after making the aforementioned changes.

After the desired adaptor output is attained (e.g. X_Y_Z1LAYER-primal.pvtu), download this output to your local machine and open it in ParaView.

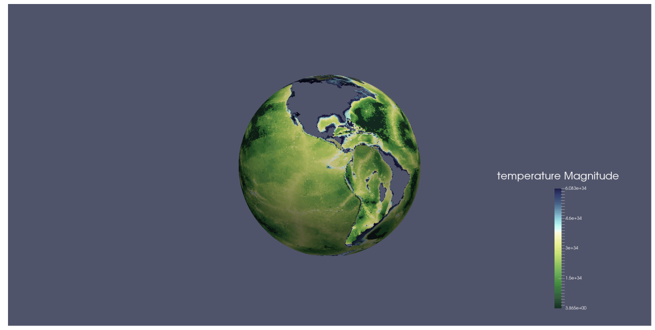
Applying Visualization Properties

Once the output is open in paraview, click the eye icon next to your file name and then click apply. A gray, hollow earth should appear in the visualization window. This grid should be missing areas that are shaped like continents as MPAS-O models the ocean not land.

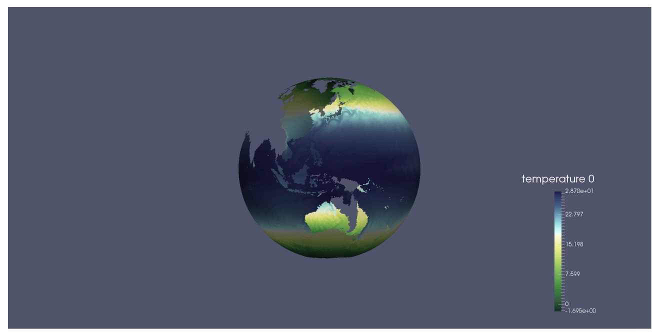
In the Cell/Point Array Status window, unselect all, then select the variable you are interested in visualizing. For the continuity of this example, I will be using Temperature. Once this is selected, click apply and the visualization should show the variable mapped in the default cool to warm colormap and a default value range.



In the top right of Paraview, there is an symbol next to the rainbow icon,  pictured here, select this tool in order to alter the colormap and data range. In order to import a color map, click the folder with a red checkmark icon and then select the option “Import”. The colormaps provided by LANL Visual Artist Francesca Samsel are located in the archive and have the .xml file extension. Alternatively, select one of the preloaded color maps. Some color maps are expected to be built into versions 4.3.2 and onward of ParaView. In this example, I selected a Blue Green Asymmetric Divergent color map.



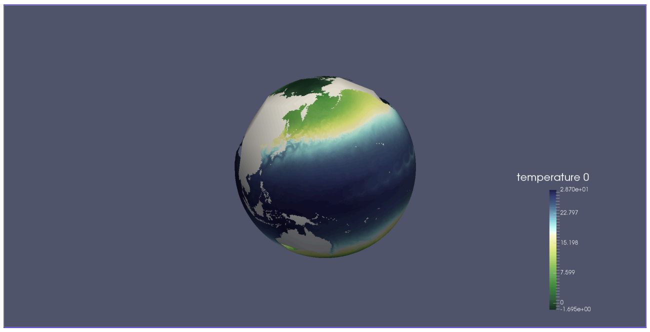
Select a the drop down to the adjacent right of the Color option marked “Magnitude” and then select a single value. For the purposes in this paper, the most appropriate value is 0. This will make the change shown below.



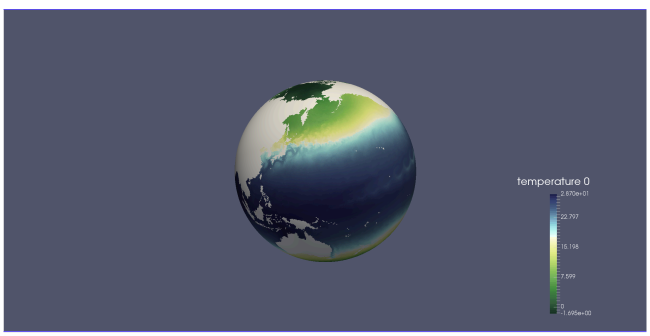
Following this step, the data range will adjust to the magnitude selected in the visualization. In this and other versions of ParaView, the setting “Vertical Level” may need to be adjusted to a value of 1 depending on the type of adaptor file used as input.

Other visualization techniques that may be of interest for the generation of MPAS-O visualizations are changing the background color, inserting a sphere, and altering the placement of the legend. Altering the legend’s placement allows for less horizontal space to be used while still generating a roughly square image should the scientist wish to later change the size of the visualization. Inserting a blank sphere allows one to better see the immediate surface and not the surface visible through the hollows. The sphere also makes, for those who are not familiar with MPAS-O, easier to distinguish that the empty areas are continental land masses, and not accidents or the like. Changing the background allows for a higher degree of visual perception when the grey is changed to a warm toned grey which has a higher degree of contrast with the cool colors of the selected color map.

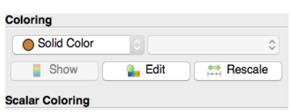
Insert a sphere by selecting Sources > Sphere then adjust the Radius option to 6371000 before selecting the apply option.



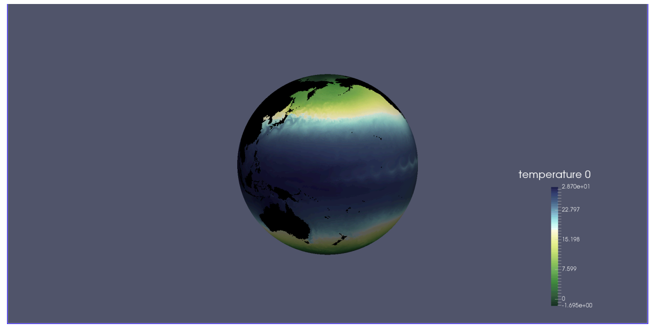
Alter the Phi Resoltuion and Theta Resolution values to 100 each to avoid rigid gaps where the sphere does not lie flat against the MPAS-O output.



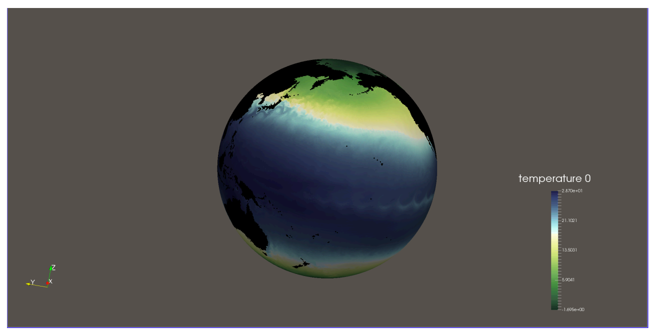
Lastly alter the color of the sphere to your color of preference.



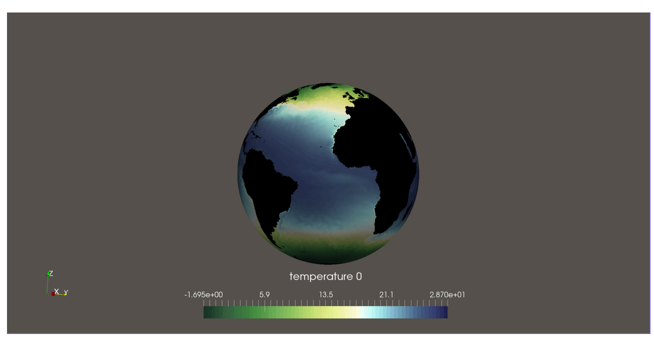
In this example, the color black is selected to communicate the regions are not of interest. Select the color tool, making sure the object “Sphere” is selected in the window that shows the objects you are visualizing. Click edit and adjust the value to 0, 0, 0 for black. Click apply.



Changing the background color is done by selecting the object X_Y_Z_1LAYER-primal_1.pvtu in the window and navigating down to the Background option. A suitable RGB value for a warm grey according to LANL visual artist F. Samsel is 86, 84, 76 and this is the value used in this example.



The legend in ParaView is moved by dragging and dropping. This change is reflected in the python code and can be altered through the code after the fact without generating a new script in ParaView Coprocessing.



Generating a python script

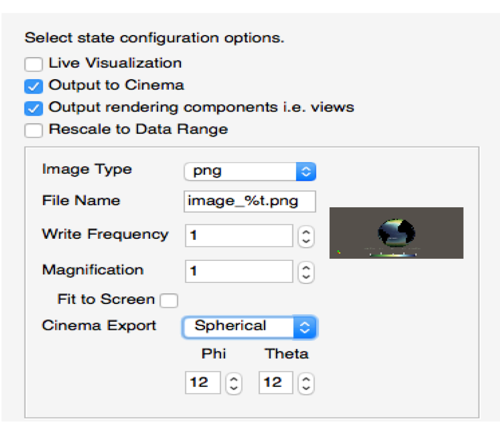
It is important to have the coprocessing plugin enabled in order to generate a Paraview python script. This is done by navigating to Tools > Manage Plugins > Catalysts Script Generator Plugin then selecting the autoload option in the check box and selecting Load Selected to load the plugin into the current window.

Certain builds of ParaView do not contain the Coprocessing Plugin and are not suitable for generating python scripts. The version recommended in this paper is chosen because it includes both the coprocessing functionality paired with cinema functionality, resulting in very few alterations needing to be made in the resultant python file.

Select CoProcessing > Export State to open the wizard. In the next window, select only the adaptor input name. Other options are available when selecting Show All Sources, however these do not need to be selected.

In the next window, the value for Simulation Name must be changed from “input” to the name matching the adaptor input you have selected. In this example, the matching value for input is X_Y_Z_1LAYER-primal. Substitute this for input and select continue.

In the next window check the box next to output to cinema and in the dropdown that appears labeled Cinema Export, select Spherical. You should see a preview of the visualization that the script will generate.



Select done and save the python file to your local machine.

It is also recommended to use the save state option to create a state file in ParaView. Should you need to make slight changes to the visualization after the fact, this will be useful.

Alterations to Python File

Some slight alterations can be made to the python script generated to make it easier to run and read.

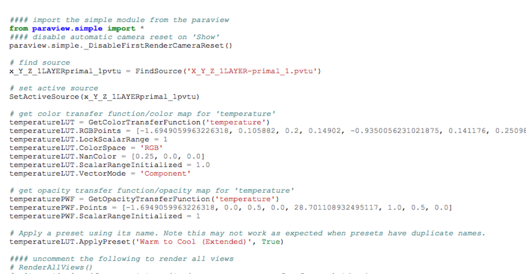
The first alteration is to remove the line

```
renderView1.AxesGrid = 'GridAxes3DActor'
```

Which is about line 28 in any given ParaView Python script. This line references undefined information for the adaptor file used in the above example.

In the event you would like to make a slight change, but do not want to repeat the visualization process in ParaView, you are able to make these changes in the code. you can open a save state in ParaView and navigate to Tools > Start Trace. Select the Fully Trace option and then select OK. This option will record any changes made to the visualization and output the appropriate python code when the trace is stopped.

For example, to change a color map without repeating the visualization process, simply repeat the aforementioned steps to select a new color map and then stop the trace. Your output will be along the lines of the following



```
#### import the simple module from the paraview
from paraview.simple import *
#### disable automatic camera reset on 'Show'
paraview.simple._DisableFirstRenderCameraReset()

# find source
x_Y_Z_1LAYERprimal_1pvtu = FindSource('X_Y_Z_1LAYER-primal_1.pvtu')

# set active source
SetActiveSource(x_Y_Z_1LAYERprimal_1pvtu)

# get color transfer function/color map for 'temperature'
temperatureLUT = GetColorTransferFunction('temperature')
temperatureLUT.RGBPoints = [-1.6949059963226318, 0.105882, 0.2, 0.14902, -0.9350056231021875, 0.141176, 0.25098,
temperatureLUT.LockScalarRange = 1
temperatureLUT.ColorSpace = 'RGB'
temperatureLUT.NanColor = [0.25, 0.0, 0.0]
temperatureLUT.ScalarRangeInitialized = 1.0
temperatureLUT.VectorMode = 'Component'

# get opacity transfer function/opacity map for 'temperature'
temperaturePWF = GetOpacityTransferFunction('temperature')
temperaturePWF.Points = [-1.6949059963226318, 0.0, 0.5, 0.0, 28.701108932495117, 1.0, 0.5, 0.0]
temperaturePWF.ScalarRangeInitialized = 1

# Apply a preset using its name. Note this may not work as expected when presets have duplicate names.
temperatureLUT.ApplyPreset('Warm to Cool (Extended)', True)

#### uncomment the following to render all views
# RenderAllViews()
# alternatively, if you want to write images, you can use SaveScreenshot(...).
```

Simply select the section of the code that is output, open the generated python file, and replace the corresponding area of code with that output by the trace function.

In order to change the title of the legend, open the python file and navigate to the following lines of code (approximately line 102):

```
temperatureLUTColorBar.Title =
'temperature'

temperatureLUTColorBar.ComponentTitle =
'0'
```

This section of the code corresponds to the title of the legend shown throughout the example of this paper. To change the title from “temperature 0” to “Temperature”, the code would appear as follows:

```
temperatureLUTColorBar.Title =
'Temperature'

temperatureLUTColorBar.ComponentTitle =
' '
```

Executing MPAS-O with Cinema

To execute the generated script, upload it to your HPC of choice into the directory of the resolution you want your cinema database in. Then create a soft link between this file and the mpas.py file. This can be done using the following command

```
ln –s GeneratedPythonScript.py mpas.py
```

In the case there is already an mpas.py soft link set up, adding the –f option to the command overwrites the existing soft link. Between multiple runs of MPAS-O with scripts that generate cinema data bases, it is a good idea to clear the contents of the image subdirectory of the directory cinema. Clearing the contents of the image subdirectory avoids cross contamination between runs, particularly those of different lengths. For example, if you generate a cinema database with 40 time steps and then proceed to run a script that outputs 37 time steps, there will be three folders from the previous mpas run that were not overwritten. If you were to run two runs that each generated 40 time steps, but your second run encountered an error or timeout midway that caused it to only generate 17 time steps, it would appear that the program ran to completion until further inspection. Because it is cumbersome to check each directory after running the executable, it is wise to simply remove the contents of that directory between runs.

To save the older as a zip file on the HPC, use the command

```
zip –r ZippedDatabase.zip image
```

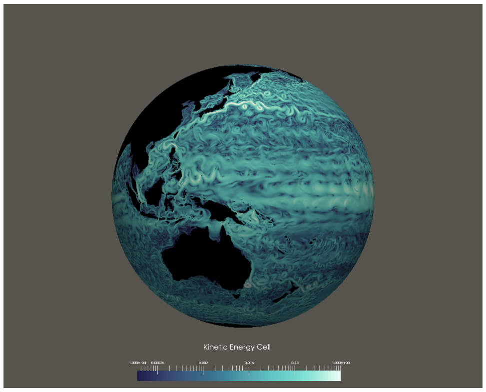
A similar option exists to recursively save the generated database as a tar file. After saving the current contents of the image subdirectory, recursively remove the folder image and then create a new directory as follows

```
rm –rf image
mkdir image
```

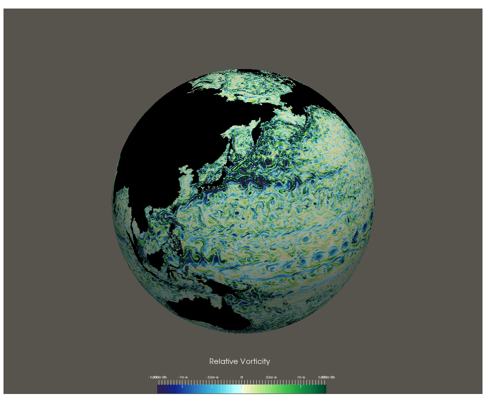
The result of executing a script to completion is a cinema database. This will appear in the HPC as a series of folders labeled 1.00000 onwards (depending on the amount of time steps you choose to generate) and an info.json file. Within the directories is a mix of folders at different values of phi and theta that contain images in .png format and associated image data.

IV. CONCLUSION

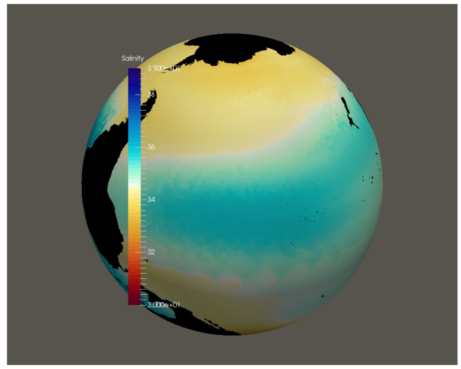
Now at the completion of this project, there are multiple scripts that generate cinema databases equivalent to the results of full post processing output from LANL's domain visual artist. These are shown below.



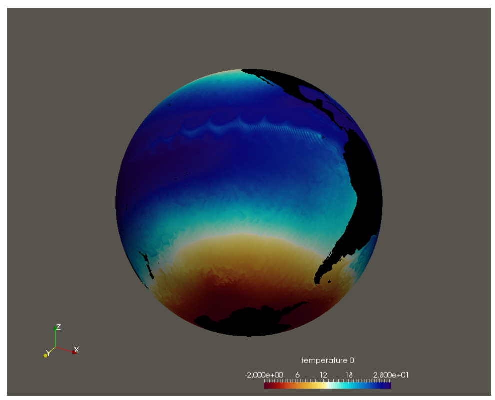
The kinetic energy script generates cinema databases the variable with a data range of 1e-04 to 1 in a linear blue color map. This color map is used for visualization in order to give an intuition for the continuity of values in the data range.



This is a visualization resulting from a script generated to run the variable Relative Vorticity in the data range -1e-05 to 1e-05 in a Blue Green divergent color map. The blue green reflects the format of the conventional Moreland cool warm color map while improving perception of features by using green. The central divergence reflects a 0 value in the center of the data range.



This results is from a script that outputs a visualization of the variable Salinity in the extended Warm Cool color map with a data range of 30 to 39.



The final result shown in above in this paper is a visualization output from a script that generates Temperature in the extended Warm Cool color map with a data range of -2 to 30 degrees Celsius.

As shown by the above visualizations, this workflow allows for the generation of cinema data bases at high resolution with a variety of variables visualized and a mix of visualization features such as choice in color map and scale placement.

REFERENCES:

1) Ahrens, J., Jourdain, S., O'Leary, P., Patchett, J., Rogers, D. H., & Petersen, M. (2014, November). An image-based approach to extreme scale in situ visualization and analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 424-434). IEEE Press.

2) Samsel, F., Petersen, M., Geld, T., Abram, G., Wendelberger, J., & Ahrens, J. (2015, April). Colormaps that Improve Perception of High-Resolution Ocean Data. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 703-710). ACM.

3) Ringler, T., Petersen, M., Higdon, R. L., Jacobsen, D., Jones, P. W., & Maltrud, M. (2013). A multi-resolution approach to global ocean modeling. *Ocean Modelling*, *69*, 211-232.